CSCI 61 – TRAN Name: _____

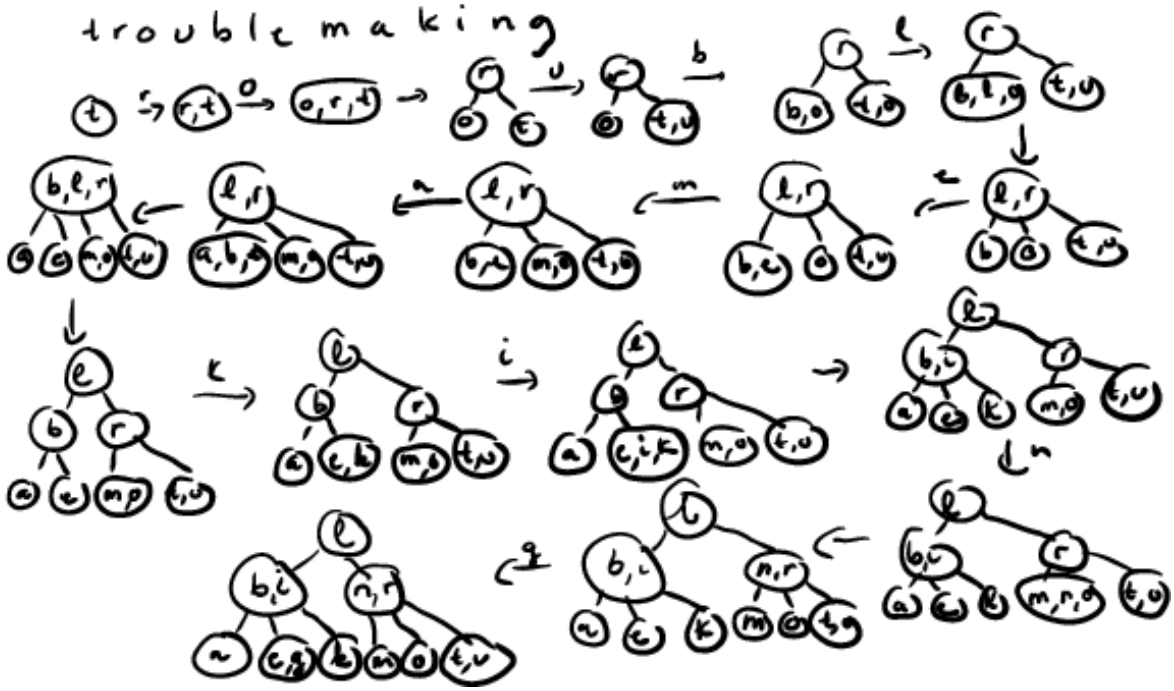
FINAL EXAM SOLUTIONS
CSCI 61-1: DATA STRUCTURES
SPRING 2016

SHOW ALL WORK!

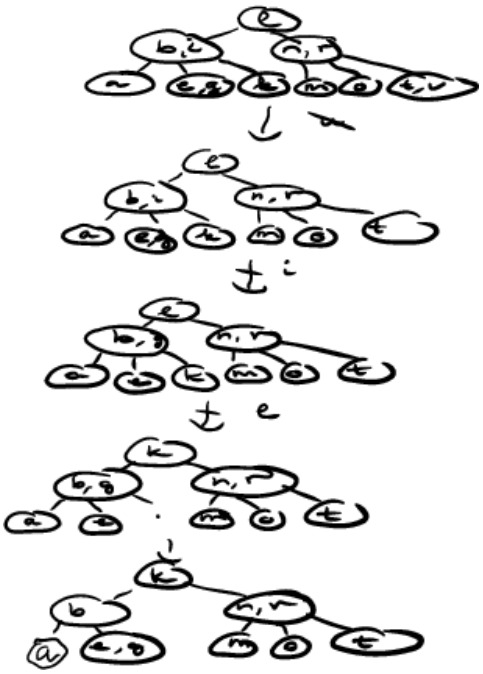
1. (10 points) Let T be an empty 2-3 tree whose keys are letters.

   (a) Insert `t r o u b l e m a k i n g` into T in the given order and show the result.

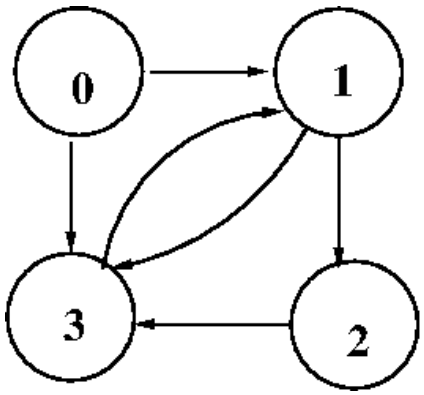
Answer:



(b) Delete `u i l` in the given order from T and show the result.
Answer:



2. (10 points) Simulate by hand a depth-first search starting at vertex 0 and always visiting the lowest-numbered neighbor first. For each vertex, give the preorder and postorder visit times. For each edge, classify it as tree, back, forward, or cross.



Answer:

```
vertex 0 (pre, post): 1, 8
vertex 1 (pre, post): 2, 7
vertex 2 (pre, post): 3, 6
vertex 3 (pre, post): 4, 5

edge (0, 1): tree
edge (0, 3): forward
edge (1, 2): tree
edge (1, 3): forward
edge (2, 3): tree
edge (3, 1): back
```

3. (10 points) Write the following member function for the class `graph`:

```
// pre: start < n()
// post: returns a list of all vertices reachable from start via some path

std::list<std::size_t> q3(std::size_t start) const;
```

Answer:

```
std::list<std::size_t> q3(std::size_t start) const
{
    assert(start < n());

    std::list<std::size_t> ans;
    std::vector<int> parent(n(), -1);

    bfs1(start, parent);
    for (std::size_t i = 0; i < n(); ++i)
       if (parent[i] != -1)
           ans.push_front(i);

    return ans;
}
```

4. (10 points) Write the following function:

```
// pre:  head_ptr points to the root of a binary search tree B;
//       1 <= i <= size of B
// post: returns the ith smallest value in B

template <class T>
T q4(const btnode<T> * head_ptr, std::size_t i);
```

Answer:

```
T q4(const btnode<T> * head_ptr, std::size_t i)
{
    assert(i >= 1 && i <= bt_size(head_ptr));

    std::size_t sl = 1 + bt_size(head_ptr->left());

    if (i == sl)
        return head_ptr->data();

    if (i < m)
        return q4(head_ptr->left(), i);

    else
        return q4(head_ptr->right(), i - sl);
}
```

5. (10 points) Write the following member function for the class `digraph`:

```
// pre: none
// post: returns true if this digraph contains a cycle of length 3;
//       false otherwise

bool q5() const;
```

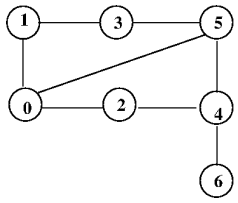
Answer:

```
bool q5() const
{
    if (n() < 3)
        return false;

    for (std::size_t i = 0; i < n(); ++i)
        for (std::size_t j = 0; j < n(); ++j)
            for (std::size_t k = 0; k < n(); ++k)
            {
                if (i == j || i == k || j == k)
                    continue;
                if (is_edge(i, j) && is_edge(j, k) && is_edge(k, i))
                    return true;
            }
    return false;
}
```

6. (10 points) Write the following member funtion for the class `graph`:

```
// pre: n() >= 2
// post: returns the longest distance between any two vertices in this graph

std::size_t q6() const;
```

For example, the longest distance between any two vertices in the graph below is 4 (between vertices 1 and 6).



Answer:

```
std::size_t q6() const
{
     std::size_t ans(0);

     for (std::size_t start = 0; start < n(); ++start)
     {
          std::queue<std::size_t> q;
          std::vector<int> d(n(), -1);
          std::size_t f;

          q.push(start);
          d[start] = 0;
          while (!q.empty())
          {
              f = q.front();
              q.pop();
              for (auto e: _v[f])
                  if (d[e] == -1)
                  {
                      q.push(e);
                      d[e] = 1 + d[f];
                  }
          }
          ans = std::max(ans, d[f]);
     }
     return ans;
}
```